

A novel method of detecting malware on Android mobile devices with explainable artificial intelligence

Seema Sachin Vanjire¹, Mohandoss Lakshmi²

¹Department of Computer Science and Engineering, Sathyabama Institute of Science and Technology, Chennai, India

²Department of Data Science and Business Systems, Faculty of Engineering and Technology, SRM Institute of Science and Technology, Kattankulathur, India

Article Info

Article history:

Received Jun 14, 2023
Revised Aug 31, 2023
Accepted Sep 28, 2023

Keywords:

Android malware
Deep learning
Explainable artificial intelligence
Malware classification
Malware detection

ABSTRACT

The increasing prevalence of malware targeting android mobile devices has raised significant concerns regarding user privacy and security. In response, effective methods for malware classification and detection are crucial to protect users from malicious applications. This paper presents an approach that leverages deep learning techniques and explainable artificial intelligence (XAI) for android mobile malware classification and detection. Convolutional neural networks (CNNs) are deep learning model that has shown impressive performance in several application areas, including image and text classification. In the context of android mobile malware, CNNs have shown promising results in capturing intricate patterns and features inherent in malware samples. By training these models on large datasets of benign and malicious applications, accurate classification can be achieved. To enhance transparency and interpretability, XAI techniques are integrated into the classification process. These techniques provide insights into the decision-making process of the deep learning models, enabling the identification of critical features and characteristics that contribute to the classification results. This research, by combining deep learning and XAI methods, presents a fresh strategy for identifying and categorizing Android malware. This research paper will focus on a fascinating CNN-based malware categorization technique.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Seema Sachin Vanjire
Department of Computer Science and Engineering, Sathyabama institute of Science and Technology
Rajiv Gandhi Salai, Jeppiaar Nagar, Chennai, India
Email: seemasvanjire@gmail.com

1. INTRODUCTION

Over the past several years, one of the primary concerns of researchers and specialists in the field of cybersecurity has been the growing frequency and sophistication of malicious software. Malware is an abbreviation that stands for malicious applications. This term refers to a broad category of potentially destructive software that is developed to take advantage of flaws in computer systems, mobile devices, and networks with the intention of wreaking havoc, stealing sensitive information, or gaining unauthorised access. Traditional methods of malware detection based on signatures were shown to be ineffective when confronted with the continually changing panorama of threats posed by malware. As a direct consequence of this, researchers working in the field of malware detection and classification began looking to machine learning and, more particularly, deep learning for more efficient solutions.

This research article, provide a deep learning-based approach to identifying and classifying android malware. Which also highlight the need for more precise results in the field of malware detection using machine

learning while presenting significance of previous work in this area. The use of a deep learning-trained convolutional neural network (CNN) algorithm is an improvement over previous work. This paper investigates a long-standing open question: how to identify novel behaviors inside malware families. The research employs a form of artificial intelligence (AI) called explainable artificial intelligence (XAI), which can justify its decisions to a human reader. The overall efficiency of the system can be improved by using XAI to learn more about android malware detection and classification. The novel contribution of this study is the application of XAI to the problem of discovering commonalities across distinct types of malware. The goal is to improve malware detection and classification by making the detection process more interpretable and transparent through the provision of justifications for the selections made by the deep learning model. The paper's next sections will detail the research methods, explaining how deep learning was implemented with CNN and how XAI was included.

While deep learning and XAI show promise in android mobile malware classification and detection, there are still challenges to overcome. Class imbalance, adversarial attacks, and real-time scalability are among the key challenges that require further investigation and development of robust solutions. The experimental findings will demonstrate the efficacy of the proposed strategy in identifying and categorizing android malware. This research, by combining deep learning and XAI methods, presents a fresh strategy for identifying and categorizing android malware. The following parts will describe the research process, provide the findings, and show why the proposed technique is important for stopping the spread of android malware.

2. LITERATURE REVIEW

Recently, deep learning techniques such as CNN have proven superior to more standard learning algorithms in a wide range. In view of this achievement, a CNN network was proposed for malware classification, along with data augmentation addresses to improve performance [1]. Bhanu *et al.* [2] provide an integrated structure for dealing with the problem of malware, with a focus on threats sent via SMS messages on android devices. SMS message processing has been implemented. Research by Susanto *et al.* [3] includes an examination of every phase of malware detection and offers an alternative taxonomy of literature concerning IoT malware detection in an effort to uncover problems and challenges in this area. Data sets from malware repositories, feature extraction techniques, and detection strategies are all discussed, as are the results of the various studies. Abubaker *et al.* [4] gives framework that uses feature selection based on an ensemble extra tree classifier approach and a machine learning classifier to examine the behavior of malware apps by evaluating permissions. When it comes to detecting and classifying malware, a method has been presented [5]. DL-droid is a deep learning system that can detect malicious android apps by methodical input generation and dynamic analysis [6]. Experiments were run on real smart phones with over 30,000 applications, both good and poor. Kim *et al.* [7] represents a malware detection system that brings together a high detection rate with an average resource need. The application programming interface call graphs of malicious programs are analyzed by MAPAS, which uses convolution neural networks. However, MAPAS uses CNN to detect shared features in malware API call graphs. Effective and efficient android malware detection using machine learning is represented in paper [8]. Deep learning for android malware detection has been suggested, and its dynamic nature makes it resistant to obfuscation [9]. Using the dynamic exploration of an application in a simulated environment, it makes advantage of the identified behavioral traits. The proposed technique is evaluated with 13533 applications representing several different domains. This method is effective, with an F-measure and a detection rate. Android application malware is also identified using a novel gated recurrent unit (GRU) and recurrent neural network (RNN) technique explained in [10]. Extracts two static attributes from Android apps using the CICAndMal2017 dataset: API calls and permissions. Malware detection and threat attribution (MDTA) is a portable framework for detecting malware [11]. Recognized threats can be identified using supervised machine learning techniques, which are also used. The most realistic and controllable strategy is the MDTA. Paper's technology used three machine algorithms to detect malware vulnerabilities in mobile app behavior. To ensure this system is capable of accurately predicting how mobile applications will behave, we put it through a battery of tests using K-nearest neighbor (KNN), Naive Bayes, and a decision tree technique [12].

3. MOBILE MALWARE DETECTION

Android anti-malware solutions such as Lookout mobiles were tested by zveloLABS. Lookout version 8.9-00fc217, a renowned android anti-virus app, was tested to check if it could detect the harmful malware programmes DroidDream. The eclipse IDE's android virtual machine plugin was used to emulate the Android smartphone environment. Then, using the disassembly/assembly technique described in the research paper, successfully changed the DroidDream binaries. This because the researchers had mentioned

that converting mobile malware could have an impact on its functionality. Because the disassembly/reassembly procedure had no effect on the code, figured DroidDream had kept its functionality. The goal of this research to see if the obfuscated version could be distinguished from the original. Lookout's mobile anti-virus logs showed that the malware that had been changed from DroidDream had been successfully detected. The logs from the app also revealed the discovery of the second sample of mobile malware, gemini, after it had been modified. This research article relied on version of Lookout version 8.7.1-edc6df5. As a result, the obfuscated DroidDream malware was able to evade detection, as evidenced by the presence of a malicious "Bowling Time" programme icon on the virtual android environment. It was possible for Lookout to modify and apply counter measures against the tested obfuscated malware quickly using reverse engineering and innovation. This is a great example of how malware development and counter measures have evolved over time using both white hat and black hat tactics. In this research mobile malware DroidDream altered into affiliate web scanners which successfully discovered it. This extra step elevated a fantastic point. Third-party scanners for mobile apps could be useful in determining the validity and legitimacy of an app. zveloLABS considers both static and behavioural approaches. In order to discover new breeds of android malware that have the same signature footprints, static analysis is insufficient. Malware detection efficiency improves dramatically when used in conjunction with behavioural detection approaches and machine learning technology. To stay competitive in today's market, established businesses must understand and protect their mobile fleet from the rapidly developing and evolving world of mobile malware. Although the vast majority of businesses are aware of mobile malware, only a small percentage understand the various types and how they infiltrate corporate devices. In order to classify ransomware using machine learning, a dynamic analysis method is presented in paper [13]. In order to track down and run malicious code, researchers have developed a sandbox system [14]. Moon *et al.* [15] proposed a framework for using semi-supervised machine learning to discover and analyze android's dynamic API calls. An example of a policy regulation taken from Kirin is as follows: the permission labels of phone and internet must not be present in an application. The most frequently requested harmful permission is access to the internet. Data prediction and classification using linear regression is discussed in paper [16].

Vanjire *et al.* [17] introduces an Android malware detection system that utilizes deep learning techniques to bolster mobile security. By employing fully-connected feedforward deep neural networks (FNN) and the autoencoder algorithm, the system achieves an impressive accuracy of 95% on a real-world dataset. These results underscore the effectiveness of deep learning FNN and autoencoder methods in detecting Android malware. Ullah *et al.* [18] presents a transparent malware detection system that integrates transfer learning and visual malware features. By leveraging both textual and visual attributes, the method aims to improve detection accuracy. Naeem *et al.* [19] assesses pertained CNN models for detecting malware from IoT device. It also explores the effectiveness of combining these CNN models with different classifiers in large-scale learning. The findings recommend utilizing a pretrained Inception-v3 CNN model, fine-tuned for improved performance, to detect IoT device malware. This approach leverages color image representations of Android Dalvik Executable Files (DEX) to enhance malware detection accuracy.

In the framework outlined in paper [20], a set of specialized detectors converts network-flow data into interpretable network events. Next, a neural network is crafted to examine this event sequence and identify specific threats, malware families, and broader malware categories. Then, the integrated-gradients technique is utilized to highlight events that together constitute the unique behavioral pattern of the threat. Kinkead *et al.* [21] presents two key contributions to enhancing malware detection in Android apps. Firstly, it proposes a new method using a CNN to pinpoint critical locations in an app's opcode sequence linked to malware. Secondly, it compares these identified locations with those highlighted by the LIME explainability method. Leveraging the Drebin benchmark dataset, the study shows strong alignment between CNN-identified malicious locations and those flagged by LIME across various malware families. The survey [22] aims to address the research gap by offering an in-depth and current overview of XAI methodologies relevant to cybersecurity challenges. Ambekar *et al.* [23] develops a novel Android malware classification framework named TabLSTMNet, leveraging recent datasets. Utilizing the NATICUSdroid and TUNADROMD datasets containing Android permissions and API attributes, TabLSTMNet employs a fusion of TabNet's attention mechanism and the long short-term memory (LSTM) architecture to differentiate between benign and malicious applications. Yan *et al.* [24] proposes an online detection system built on FPGA technology, aimed at enabling real-time detection in high-speed network environments. This system utilizes a rule tree approach, which simplifies the challenge of integrating a deep neural network into FPGA. Smmarwar *et al.* [25] presents XAI-AMD-DL, a hybrid Android malware detection (AMD) system that integrates CNN and Bi-GRU architectures within an XAI on CICAndMal2019 Android malware dataset.

4. MOBILE MALWARE DETECTION TECHNIQUE

Mobile malware detection and other security vulnerabilities detection approaches have varied degrees of effectiveness and weakness. Analyses that are performed in a static environment an application's static analysis can detect dangerous features or faulty code portions without executing it, saving time and money. Figure 1 shows techniques for examination when doubtful applications are checked to detect security issues.

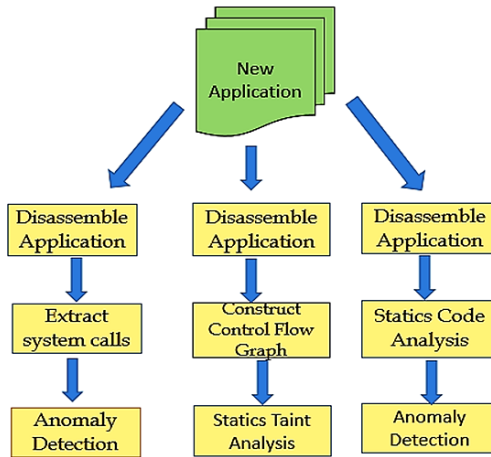


Figure 1. Static anomaly detection techniques

Figure 1 depicts a malware discovery method for operating system that has been presented. System feature extraction is used to deconstruct the mobile app and extract system calls. Then, centroid machine, a lightweight clustering approach (anomaly detection), is used to determine whether the mobile app is harmful or not. Further, it's difficult to say how well this strategy will work with less well-known ones. Figure 1 shows how to use the previously described PiOS technique to perform analysis for iOS application binaries. iOS reviews mobile application and produces a control flow diagram. This procedure needs analysing sensitive sources from phone as contact book, GPS time, and other data. Any delicate information sent from the source to sync without informing the user will be detected by dataflow analysis, resulting in security breaches. Figure 1 depicts a proposed android malware detection method. In static code analysis software examines Java source code from the installation image of the application. Despite the fact that this technique was verified on 1,100 mobile applications. To do dynamic analysis instead of static testing, researchers run the mobile application on a virtual machine or emulator. This allows them to track the app's dynamic behaviour. Dynamic analysis is used system call tracing also in taint tracking. TaintDroid used in system-wide taint tracking for android.

There are four granularities of taint propagation, as shown in Figure 2(a) variable, method, message, and the Dalvik virtual machine as well as on a filesystem level. Any uncertain data originating from sensitive sources will be marked with taint tracking to prevent unauthorised access. Location, microphone, camera, and unique phone identifiers are all examples of sensitive mobile data. By modifying the library loader in this way, we can ensure that untrusted applications are unable to execute native methods without first invoking the virtual machine. After that, sensitive data leaking is investigated by doing a dynamic analysis on the affected files.network interface—a conduit via which corrupted information can be flushed before transmission.

Many third party android applications tested with TaintDroid indicated that shared user location with phone unique id. TaintDroid, on the other hand, may produce false negatives and positives, and it ignores additional vulnerabilities in favour of just monitoring dataflow. With the android application sandbox technologies, android applications can undergo with a two-step analysis process. As shown in Figure 2(b), static and dynamic evaluations are carried out in offline mode using a sandbox. Static analysis is used to disassemble an application's binary image, and the stripped code is examined for any unusual patterns. The binary is run at android emulator then system calls are analysed dynamically. While android monkey was used to generate inputs, real-world testing was more effective. Furthermore, this method has not been tested against malware that uses polymorphic encryption. Permissions assessment for a given application are critical in mobile apps because they let users know what the app's intentions are and what it's doing behind the scenes. Because permissions on smartphones are explicitly established, application developers must obtain the necessary permissions. Application vulnerabilities can occur when writers deliberately obfuscate about permissions used in their applications.

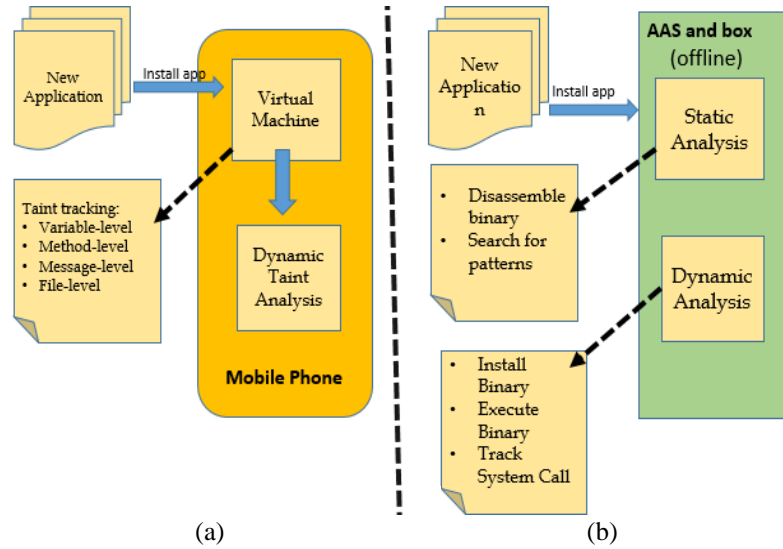


Figure 2. Dynamic anomaly analysis: (a) system-wide and (b) sandbox-based

Figure 3 depicts the Kirin android app certification. Checks the application's permissions during installation. When a programme is installed, the security settings of the user are extracted and compared to the security policy rules. If an application does not comply with Kirin's security policy, it will be deleted or a warning will be sent to the user's device. Moon *et al.* [15] proposed a framework for using semi-supervised machine learning to discover and analyze android's dynamic API calls. An example of a policy regulation taken from Kirin is as follows: the permission labels of phone and internet must not be present in an application. The most frequently requested harmful permission is access to the internet.

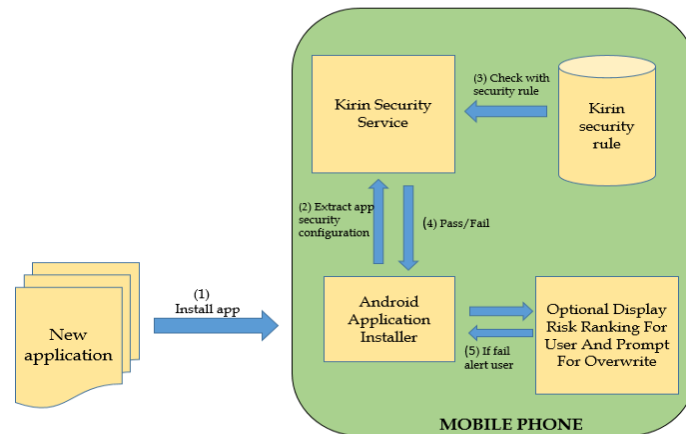


Figure 3. Kirin android application certification

5. SYSTEM PROCESS METHOD

Experimental evaluations on a comprehensive dataset of android mobile applications represents the proposed approach. The deep learning models trained with XAI techniques achieved high classification accuracy and provided interpretable explanations for the decision-making process. This enables users and security analysts to better understand the basis for malware detection, improving trust and aiding in the decision-making process. A dataset of 833,000 binary samples (both clean and malware) from various malware families, compilers, and "first-seen" time periods was collected for the malware detection investigations. Sanity checks were performed to exclude corrupt or excessively large or small samples from the experiment. The raw bytes from the samples that passed the sanity checks were extracted and used for further experiments. Deep learning model training: a CNN was trained using the raw byte data from the training set. The CNN was trained to classify the samples as either clean or malware. Evaluation and

performance analysis: three tests were conducted to evaluate the performance of the trained CNN model: a high receiver operating curve (ROC) score of 0.9953 was obtained, indicating excellent performance. Duplicate removal: duplicate raw byte entries were found after extracting them from the 833,000 unique samples due to malware families employing hash-busting and polymorphism techniques. The duplicated entries were removed in the second experiment, resulting in a reduced dataset of 262,000 samples. The performance of the model was still within an acceptable confidence range. Malware classification: in the third experiment, malware classification was attempted based on different malware families. A subset of samples was taken from the dataset, with specific labels assigned to clean samples, malware families, and others. The remaining 20,000 samples were divided into 11 categories based on different compilers and packagers. Training and testing: the dataset was randomly divided again into training and test sets using an 80/20 split. The performance of the model was evaluated using precision, and a test precision of 0.9700 was achieved in this experiment. The training and testing process took 26 minutes using a single GPU.

The method outlined in Figure 4 demonstrates the systematic process followed for android mobile malware classification and detection using a deep learning algorithm. The collection and preprocessing of a large-scale dataset, training a CNN model, evaluating performance, handling duplicate entries, and classifying malware based on different families were key steps in this method. The achieved results and performance metrics were derived from the rigorous application of this method.

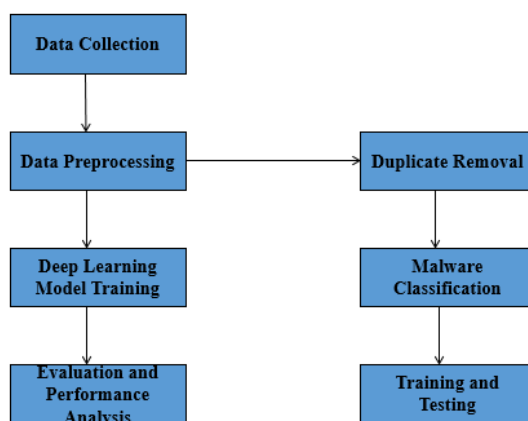


Figure 4. System process model

6. RESULTS ANALYSIS

We gathered 833,000 binary samples (dirty and clean) from a wide variety of families, compilers, and "first-seen" times for our malware detection studies. Many samples were from the same families, but the packers and obfuscators they used were all different. We performed sanity checks on the samples to make sure we weren't using any that were corrupted or were either too big or too small for the experiment. We performed a number of experiments using the unprocessed bytes from a sample that made it past our sanity check. A random 80/20 split was used to separate the data into training and testing sets. We ultimately relied on this data set to conduct the three analyses. Area under the ROC of 0.9953 was achieved by feeding the CNN raw bytes from a training set of 833,000 samples. Here, after extracting them from the initial set of 833,000 unique samples, duplicate raw byte entries were discovered. Malware groups that used hash-busting to exploit polymorphism played a significant role in this. In our second trial, we replicated the extracted raw byte entries. The raw byte input vector had dropped to a meager 262,000 samples. The evaluated area was found to have a ROC coefficient of 0.9920, well within the bounds of respectable certainty. Third, we looked for ways to group malware into its many families. The original set of samples was reduced to a total of 130,000, with the first sample marked as clean, the second through ninth as malware families, and the tenth as others. The remaining 20,000 samples were then separated into the following 11 groups: there are 11 bins total, and each one contains code snippets generated by a different compiler or packager. Again, randomly split the data into a training and test set, but this time split it 80/20. An experimental recision of 0.9700 was achieved. Training and testing on a single GPU took 26 minutes.

T-SNE and PCA are used to visually portray and explain the data both before and after the CNN training process. To get a better grasp on how CNN training works, we analyzed it graphically. After training, CNN is able to extract meaningful representations for capturing the attributes of different forms of malware, as seen in separate clusters. Since most groups were clearly distinguished, we reasoned that the method could

function as a multi-class classifier. After then, we analyzed CNN's decisions with XAI. Therefore, CNN places a premium on those details when making choices. Were interested in learning more about the bits that made a big difference in the final verdict, and so personally assessed a few cases. To get to this end, result enlisted the aid of a human specialist evaluate if the red-highlighted bytes belonged to a known malware family. The CNN's ability to learn and identify useful patterns that people or other forms of automation would overlook is demonstrated by its ability to link these bytes to virus classification. These research' lack of sophistication in uncovering novel patterns of interest is not at all indicative of the CNN's ineffectiveness.

7. CONCLUSION

By integrating deep learning methods with XAI, this study introduces a fresh strategy for android mobile malware categorization and detection. CNN models combined with XAI techniques allow for precise categorization with full explanations of the reasoning behind the results. This study aids current initiatives to reduce Android mobile malware threats, improving safety and privacy in the dynamic mobile app ecosystem. This research paper will focus on a fascinating CNN-based malware categorization technique. Using the CNN raw byte model, malware can be classified from beginning to end. CNN has the potential to be used as a feature extractor to improve existing features. With enough time and resources, the CNN raw byte model could detect hazard families before other vendors and uncover previously unknown threats. Wireless-enabled personal digital assistants (PDAs) are targets for mobile malware, which can cause the system to crash and compromise confidential data. Wireless phones and PDA networks have risen in popularity and sophistication as a result, making it more difficult to keep them safe from viruses and other types of malwares. It also provides insights on CNN judgments and aids in the identification of intriguing patterns across malware families using XAI.




REFERENCES

- [1] A. D. Jasim and R. I. Farhan, "Intelligent malware classification based on network traffic and data augmentation techniques," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 30, no. 2, pp. 903-908, May 2023, doi: 10.11591/ijeecs.v30.i2.pp903-908.
- [2] J. S. Bhanu, J. K. R. Sastry, and T. C. Reddy, "Protecting Android based applications from malware affected through SMS messages," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 22, no. 3, pp. 1580-1589, Jun. 2021, doi: 10.11591/ijeecs.v22.i3.pp1580-1589.
- [3] Susanto, M. A. S. Arifin, D. Stiawan, M. Y. Idris, and R. Budiarto, "The trend malware source of IoT network," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 22, no. 1, pp. 450-459, Apr. 2021, doi: 10.11591/ijeecs.v22.i1.pp450-459.
- [4] H. Abubaker, A. Ali, S. M. Shamsuddin, and S. Hassan, "Exploring permissions in android applications using ensemble-based extra tree feature selection," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 19, no. 1, pp. 543-552, Jul. 2020, doi: 10.11591/ijeecs.v19.i1.pp543-552.
- [5] E. Ko, J. Kim, Y. Ban, H. Cho, and J. H. Yi, "ACAMA: Deep Learning-Based Detection and Classification of Android Malware Using API-Based Features," *Communication Security in Socialnet-Oriented Cyber Spaces 2021*, vol. 2021, Dec. 2021, doi: 10.1155/2021/6330828.
- [6] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "DL-Droid: Deep learning based android malware detection using real devices," *Computers & Security*, vol. 89, Feb. 2020, doi: 10.1016/j.cose.2019.101663.
- [7] J. Kim, Y. Ban, E. Ko, H. Cho, and J.H. Yi, "MAPAS: a practical deep learning-based android malware detection system," *International Journal of Information Security*, vol. 21, pp. 725-738, 2022, doi: 10.1007/s10207-022-00579-6.
- [8] R. Srinivasan, S. Karpagam, M. Kavitha, and R. Kavitha, "An Analysis of Machine Learning-Based Android Malware Detection Approaches," *Journal of Physics: Conference Series*, vol. 2325, Jun. 2022, doi: 10.1088/1742-6596/2325/1/012058.
- [9] V. Sihag, M. Vardhan, P. Singh, and G. Choudhary, "De-LADY: Deep learning based Android malware detection using Dynamic features," *Journal of Internet Services and Information Security (JISIS)*, vol. 11, no. 2, pp. 34-45, May 2021, doi: 10.22667/JISIS.2021.05.31.034.
- [10] O. N. Elayan and A. Mustafa, "Android Malware Detection Using Deep Learning," *Procedia Computer Science*, vol. 184, pp. 847-852, Jan. 2021, doi: 10.1016/j.procs.2021.03.106.
- [11] S. S. Vanjire and M. Lakshmi, "MDTA: A New Approach of Supervised Machine Learning for Android Malware Detection and Threat Attribution Using Behavioral Reports," vol. 68, pp. 147-159, 2022, doi: 10.1007/978-981-16-1866-6_10.
- [12] S. Vanjire and M. Lakshmi, "Behavior-Based Malware Detection System Approach For Mobile Security Using Machine Learning," *2021 International Conference on Artificial Intelligence and Machine Vision (AIMV)*, Gandhinagar, India, 2021, pp. 1-4, doi: 10.1109/AIMV53313.2021.9671009.
- [13] E. B. Karbab and M. Debbabi, "Automatic investigation framework for android malware cyber-infrastructure," *arXiv*, pp. 1-12, 2018, doi: 10.48550/arXiv.1806.08893.
- [14] I. Paul, F. Irolla, and E. Filiol, "Glassbox: Dynamic Analysis Platform for Malware Android Applications on VI Real Devices," *arXiv Preprint*, pp. 1-11, 2016, 10.48550/arXiv.1609.04718.
- [15] D. Moon, H. Im, J. D. Lee, and J. H. Park, "MLDS: Multi-layer defense system for preventing advanced persistent threats," *Symmetry*, vol. 6, no. 4, pp. 997-1010, 2014, doi: 10.3390/sym6040997.
- [16] T. Akhtar, B. B. Gupta, and S. Yamaguchi, "Malware propagation effects on SCADA system and smart power grid," *2018 IEEE International Conference on Consumer Electronics (ICCE)*, Las Vegas, NV, USA, 2018, pp. 1-6, doi: 10.1109/ICCE.2018.8326281.
- [17] S. Vanjire and M. Lakshmi, "FNN and Auto Encoder Deep Learning-Based Algorithm For Android Cyber Security," *International Journal Of Recent Technology And Engineering (IJRTE)*, no. 8, no. 5, pp. 3292-3296, 2020, doi:




- 10.35940/ijrte.E6454.018520.
- [18] F. Ullah, A. Alsirhani, M. M. Alshahrani, A. Alomari, H. Naeem, and S. A. Shah, "Explainable Malware Detection System Using Transformers-Based Transfer Learning and Multi-Model Visual Representation," *Sensors*, vol. 22, no. 8, pp. 1-22, 2022, doi: 10.3390/s22186766.
 - [19] H. Naeem, B. M. Alshammari, and F. Ullah, "Explainable Artificial Intelligence-Based IoT Device Malware Detection Mechanism Using Image Visualization and Fine-Tuned CNN-Based Transfer Learning Model," *Computational Intelligence & Neuroscience*, vol. 2022, pp. 1-17, 2022, doi: 10.1155/2022/7671967.
 - [20] P. Prasse, J. Brabec, J. Kohout, M. Kopp, L. Bajer, and T. Scheffer, "Learning explainable representations of malware behavior," *Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track: European Conference, ECML PKDD 2021*, Bilbao, Spain, 2021, pp. 53-68, doi: 10.1007/978-3-030-86514-6_4
 - [21] M. Kinkad, S. Millar, N. McLaughlin, and P. O'Kane "Towards explainable CNNs for Android malware detection," *Procedia Computer Science*, vol. 184, pp. 959-965, 2021, doi: 10.1016/j.procs.2021.03.118.
 - [22] Z. Zhang, H. A. Hamadi, E. Damiani, C. Y. Yeun and F. Taher, "Explainable Artificial Intelligence Applications in Cyber Security: State-of-the-Art in Research," in *IEEE Access*, vol. 10, pp. 93104-93139, 2022, doi: 10.1109/ACCESS.2022.3204051
 - [23] N. G. Ambekar, N. N. Devi, S. Thokchom, and Yogita, "TabLSTMNet: enhancing android malware classification through integrated attention and explainable AI," *Microsystem Technologies*, pp. 1-19, 2024, doi: 10.1007/s00542-024-05615-0.
 - [24] A. Yan *et al.*, "Effective detection of mobile malware behavior based on explainable deep neural network," *Neurocomputing*, vol. 453, pp. 482-492, 2021, doi: 10.1016/j.neucom.2020.09.082.
 - [25] S. K. Smmarwar, G. P. Gupta and S. Kumar, "XAI-AMD-DL: An Explainable AI Approach for Android Malware Detection System Using Deep Learning," *2023 IEEE World Conference on Applied Intelligence and Computing (AIC)*, Sonbhadra, India, 2023, pp. 423-428, doi: 10.1109/AIC57670.2023.10263974.

BIOGRAPHIES OF AUTHORS



Seema Sachin Vanjire    is research scholar at the Department of Computer Science Engineering, Sathyabama Institute of Science and Technology, Jeppiaar Nagar, Rajiv Gandhi Salai Chennai. Seema graduated with a first-class honours B.Eng. degree in Computer Engineering from Shivaji University, Maharashtra, India, in 2005, and an M.Tech. in Computer Science and Engineering from Vishwakarma Institute of Technology Pune University, India in 2013. Her research interests are primarily in the area of wireless communications and networks as well as machine learning, deep learning, and artificial intelligence. She can be contacted at email: seemasvanjire@gmail.com.



Dr. Mohandoss Lakshmi    presently working as Professor and Head, Department of Data Science and Business Systems, SRM Institute of Science and Technology and she has an experience of more than 25 years in Teaching. She previously worked as Professor in Saveetha University, as Principal in Sri Krishna College of Technology, Coimbatore and as Professor and Dean, School of Computing, Sathyabama University, Chennai. She has completed her B.E. (Computer Science and Engineering) from Bharathidasan University and M.E. (Computer Science and Engineering) from Madras University and Ph.D. from Sathyabama University in the area of Wireless Ad Hoc networks. She has published more than 60 papers in International Journals and Conferences. She has organized many workshops to students, faculty and research scholars and also organized many national conference and international conferences. She is currently guiding eight research scholars and 8 have completed Ph.D., under her guidance. She has completed a cumulative research project funded by Indira Gandhi Center for Atomic Research (IGCAR), Kalpakkam titled "Development of Data mining tools for sequencing assembly/disassembly of the Fuel Transfer Arm for PFBR and identification of failure modes during operations". She is a DST FIST Project coordinator and got a funding of 57 lakhs from DST to establish wireless sensor and IoT Lab. She was the founder coordinator for Student Development Cell at Sathyabama University. She has received BEST TEACHER Award five times in the year 2003, 2007, 2009, 2012, and 2014 from Sathyabama University. She can be contacted at email: laks@icadsindia.com.